Locating the eigenvalues for graphs of small clique-width

Martin Fürer
Institut für Theoretische Informatik, ETHZ
Visiting from Pennsylvania State University

Joint work with David P. Jacobs an Vilmar Trevisan

May 19, 2016

Tree-width

- The subject started with Gaussian elimination for sparse matrices.
- Want to minimize fill-in.
- Intensive study of tree-width by Robertson and Seymour.
- Existence of polynomial time algorithms proved, without providing such algorithms.
- Minor closed classes of graphs. (A minor is obtained by vertex and edge deletions and edge contractions.)

Why tree-width?

- Many combinatorial graph problems are NP-hard.
- Usually, they are easy for trees.
- One wants to extend feasibility to a somewhat more general classes of graphs.
- The tree-width measures similarity to trees.
- Low tree-width usually implies efficient algorithms.

Tree-width

- Defined with the help of a tree-decomposition.
- Tree-width tw(G): Smallest k, having a tree decomposition with all bags of size $\leq k + 1$.
- There are many efficient algorithms for graphs of small tree-width.
- Courcelles (1993) theorem: O(f(k) n) time algorithms (linear FPT) for all Monadic Second Order properties of vertices and edges.

Wanted: More powerful graph classes

- Bounded tree-width graphs are sparse.
- Most problems are easy for simple dense graphs like K_n or K_{pq}.
- Extend to a nice class?
- Intuitive property: Easily formed by adding all edges between two sets of vertices.
- Clique-width measures the complexity of such constructions.
- It is a more recently defined width parameter.

k-expression defining a labeled graph

- Label set $= [k] = \{1, 2, \ldots, k\}$.
- Operations:

$i(v)$	create vertex v with label i.
$\eta_{i,j}$	create all edges between the vertices labeled i and the vertices labeled j (for $i \neq j$).
$\rho_{i \to j}$	change all labels i to j.
$\oplus$	disjoint union of two graphs

- At the end, forget the labels.
- Clique-width $cw(G)$ = smallest number of labels that can produce G.
- E.g., a clique of any size has clique-width 2.

Locating eigenvalues

- The number of eigenvalues grater than c is equal to the number of positive eigenvalues of $B = A - c\,I$.
- Thus the number of eigenvalues in an interval can be determined by computing the number of positive and negative eigenvalues of two matrices.
- For regular P, the matrix $P^T BP$ is congruent to B.
- Sylvester: Congruent matrices have the same number of positive (negative) eigenvalues.
- Transform B into a congruent diagonal matrix.

The result

Theorem

For graphs of bounded clique-width with a given k-expression, the number of eigenvalues in an interval can be computed in linear time.

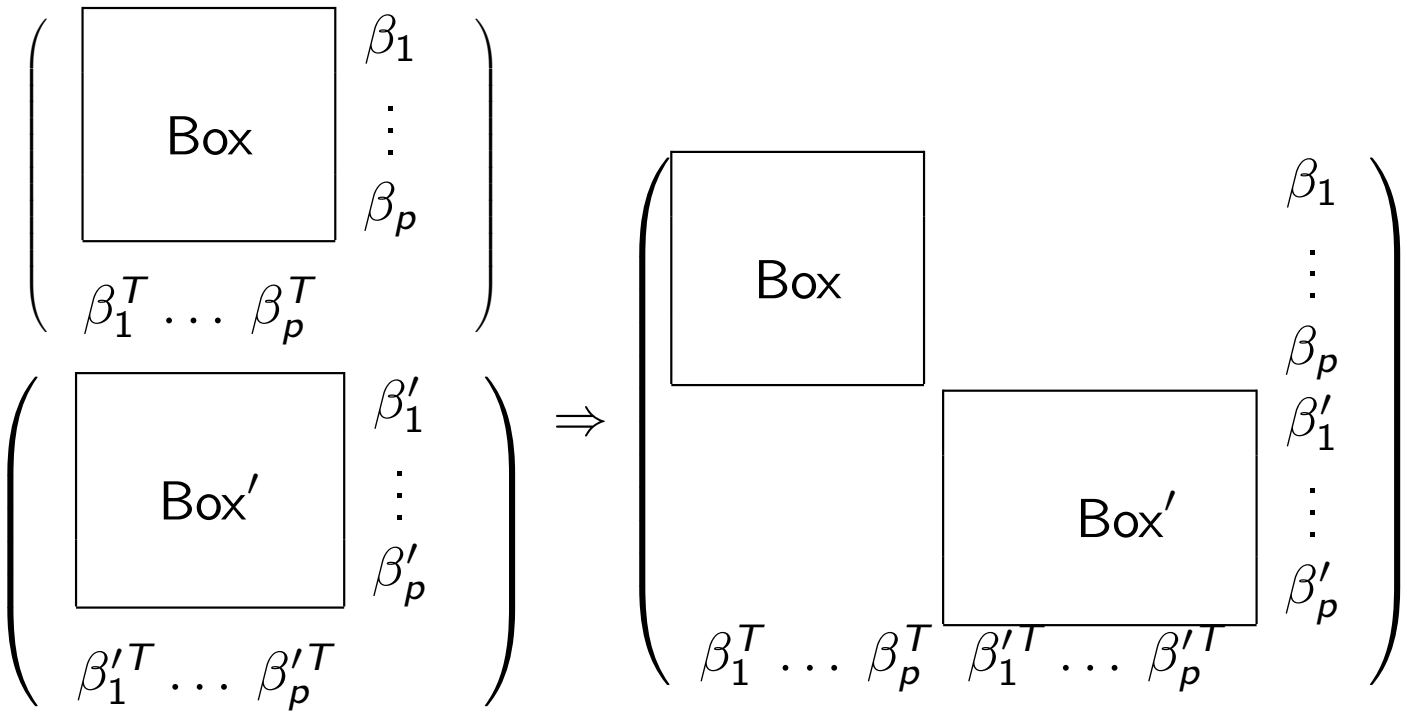
The shape of the matrix B_q (congruent to B_0)

$$B_q = \begin{pmatrix} b_{-q+1} & & & & \\ & \ddots & & 0 & \\ & & b_0 & & \\ & 0 & & \boxed{\begin{matrix} b_{11} & \dots & b_{1p} \\ \vdots & & \vdots \\ b_{p1} & \dots & b_{pp} \end{matrix}} & \begin{matrix} \beta_1 \\ \vdots \\ \beta_p \end{matrix} \\ & & & \begin{matrix} \beta_1^T & \dots & \beta_p^T \end{matrix} & \beta \end{pmatrix}$$

Methodology

- For clique-width k, we focus on a small $p \times p$ submatrix (square box), where $p \leq 2k$.
- Idea: We aim at the submatrix containing the rows and columns of one representative vertex per label.
- The algorithm uses the parse tree of a k-expression, and performs matrix operations corresponding to the operations in the k-expression.
- We only use congruence operations that add a multiple of a row and column to an other row and column respectively.

The simple operations



- $i(v)$: Focus on the 1×1 square box of vertex v.
- $\oplus$: Focus on square box around the two given square boxes.
- $\eta_{i,j}$: Insert some 1's in the square box.
- $\rho_{i \to j}$ record the new label.

Two vertices, say 1 and i have the same label: First step

$$\left(\begin{array}{|ccc|c} \multicolumn{3}{|c|}{\multirow{3}{*}{}} & \beta_1 \\ a_0 & b_0 & \cdots & \beta_2 \\ b_0 & c_0 & \cdots & \vdots \\ \multicolumn{3}{|c|}{} & \beta_p \\ \hline \multicolumn{4}{c}{\beta_1^T \beta_2^T \ldots \beta_p^T} \end{array}\right) \Rightarrow \left(\begin{array}{|ccc|c} \multicolumn{3}{|c|}{\multirow{3}{*}{}} & \vec{0} \\ a & b & \cdots & \beta_2 \\ b & c & \cdots & \vdots \\ \multicolumn{3}{|c|}{} & \beta_p \\ \hline \multicolumn{4}{c}{\vec{0}^T \beta_2^T \ldots \beta_p^T} \end{array}\right)$$

- In pictures, we omit the already diagonalized part for simplicity.
- The box represents the interesting submatrix.
- $\vec{0}$ and the β_i are (currently uninteresting) row-vectors.
- We show the first step when vertices 1 and i have the same label, implying $\beta_1 = \beta_i$.
- Subtract row/column i from row/column 1.

Case distinction

$$\begin{pmatrix} \boxed{\begin{matrix} a & b & \cdots \\ b & c & \cdots \\ \vdots & \vdots & \end{matrix}} & \begin{matrix} \vec{0} \\ \beta_2 \\ \vdots \\ \beta_p \end{matrix} \\ \vec{0}^T \; \beta_2^T \; \ldots \; \beta_p^T & \end{pmatrix}$$

Case 1: $a \neq 0$: Clear first row and column except for a in diagonal.

Case 2: Top row is all 0: 0 in diagonal.

Case 3: $a = 0$, top row not all 0, w.l.o.g. $b \neq 0$.

The interesting Case 3

$$\left(\begin{array}{ccc|c} 0 & b & \cdots & \vec{0} \\ b & c & \cdots & \beta_2 \\ \vdots & \vdots & & \vdots \\ & & & \beta_p \\ \hline \multicolumn{3}{c|}{\vec{0}^T\ \beta_2^T \ \ldots\ \beta_p^T} & \end{array}\right) \Rightarrow \left(\begin{array}{ccc|c} 0 & b & \vec{0} & \vec{0} \\ b & c & \cdots & \beta_2 \\ \vec{0}^T & \vdots & & \vdots \\ & & & \beta'_p \\ \hline \multicolumn{3}{c|}{\vec{0}^T\ \beta_2^T \ \ldots\ \beta_p'^T} & \end{array}\right)$$

- Only Case 3 ($a = 0$) needs more work.
- Add appropriate multiples of row/column 2 to clear column/row 1 after position 2.
- For some constants $c_3, \ldots, c_p$, $(\beta_3, \ldots, \beta_p)$ gets replaced by $(\beta'_3, \ldots, \beta'_p) = (\beta_3 + c_3\beta_2, \ldots, \beta_p + c_p\beta_2)$.

The interesting Case 3

$$\begin{pmatrix} \boxed{\begin{matrix} 0 & b & \vec{0} \\ b & c & \cdots \\ \vec{0}^T & \vdots & \end{matrix}} & \begin{matrix} \vec{0} \\ \beta_2 \\ \vdots \\ \beta_p' \end{matrix} \\ \begin{matrix} \vec{0}^T & \beta_2^T & \ldots & \beta_p'^T \end{matrix} & \end{pmatrix} \Rightarrow \begin{pmatrix} \boxed{\begin{matrix} -b & 0 & \vec{0} \\ 0 & b & \vec{0} \\ \vec{0}^T & \vec{0}^T & \end{matrix}} & \begin{matrix} \vec{0} \\ \vec{0} \\ \vdots \\ \beta_p' \end{matrix} \\ \begin{matrix} \vec{0}^T & \vec{0}^T & \ldots & \beta_p'^T \end{matrix} & \end{pmatrix}$$

- Add multiples of row/column 1 to clear column/row 2 after position 2.
- Add $(1 - c/b)/2$ times row/column 1 to row/column 2. Now c is b.
- Add multiples of row/column 1 to all rows/colums after 2 to clear column/row 2 after position 2.
- Subtract row/column 2 from row/column 1.
- Now the first two rows and columns are diagonalized.

Complication

$$B_q = \begin{pmatrix} \boxed{\begin{matrix} -b & 0 & \vec{0} \\ 0 & b & \vec{0} \\ \vec{0}^T & \vec{0}^T & \end{matrix}} & \begin{matrix} \vec{0} \\ \vec{0} \\ \vdots \\ \beta'_p \end{matrix} \\ \vec{0}^T \; \vec{0}^T \; \ldots \beta'^T_p & \end{pmatrix}$$

- β_2 has disappeared as its own row vector. It is unrepresented by the matrix B_q
- It shows up in other rows as $\beta'_i = \beta_i + c_i\beta_2$ for $i > 2$.
- Thus, as more entries of β_2 get known, β'_i has to be updated.
- When another vertex later shows up with label 2, and thus with row vector β_2, then β_i can be reconstructed by subtracting $c_i\beta_2$ from β'_i.

More complications

- At the beginning, we assumed two vertices, 1 and i have the same label and hence $\beta_1 = \beta_i$. Thus $\vec{0}$ has been created by subtraction.
- In reality, we subtact β_i' from β_i'.
- Instead of $\vec{0}$, we obtain a linear combination of the vectors of the unrepresented vertices.
- Instead of aiming at having just one row (and column) per label, we represent up to u additional rows, where u is the number of unrepresented vertices (which is less than the clique-width k).
- The additional rows are intuitively close to $\vec{0}$. They are just linear combinations of the vectors of the unrepresented vertices.
- If one more additional row is obtained, then $\vec{0}$ can be linearly combined, and we can continue as in the typical case where the earlier Case 3 never happened.

The end

Thank you!